Web Application's Security Demystified

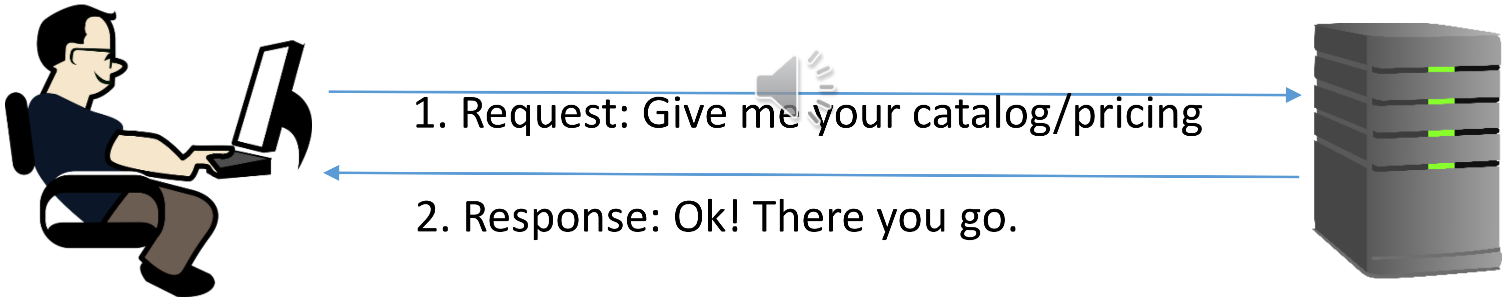
Learn some very basic concepts of securing web applications without using technical jargons.

By: Ashish Kumar
akumar@ipcolony.com

Web (http) protocol

- It is one of the easiest protocol and it is "**stateless**." The receiver has no idea, what the user did before.
- It has not changed since Al Gore invented the internet.
- Technocrats and software companies have added unnecessary jargons and terminology, making it harder to understand.

Web protocol – Simplistic use case



1. Request: Give me your catalog/pricing

2. Response: Ok! There you go.

User

Web Server

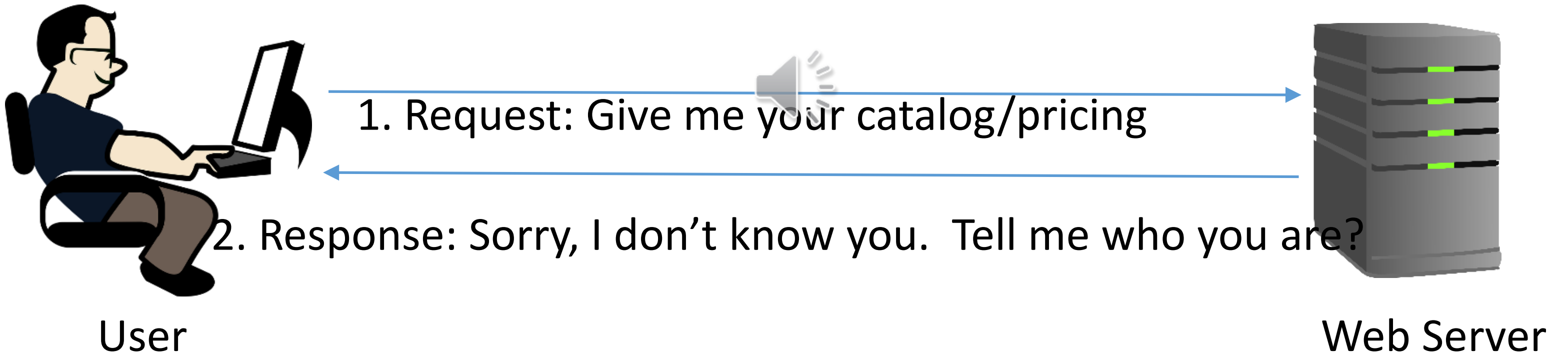
Analysis of the previous slide

- There is no security.
- Web server delivers the catalog of the products as requested.

What if the pricing varied from user to user?

After all, Boeing being our best customer gets the best prices.

Web protocol – Let us start again



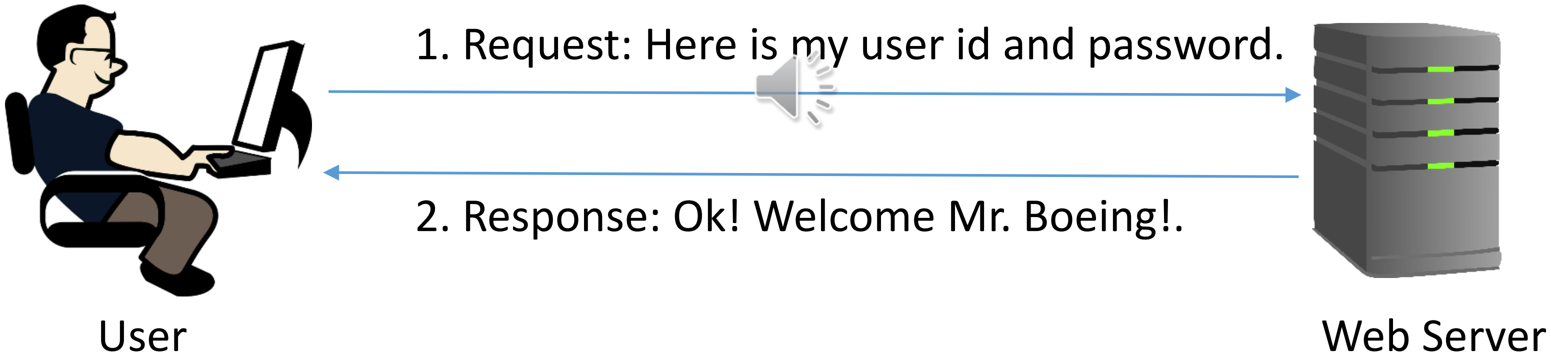
1. Request: Give me your catalog/pricing

2. Response: Sorry, I don’t know you. Tell me who you are?

User

Web Server

Web protocol – Login completed



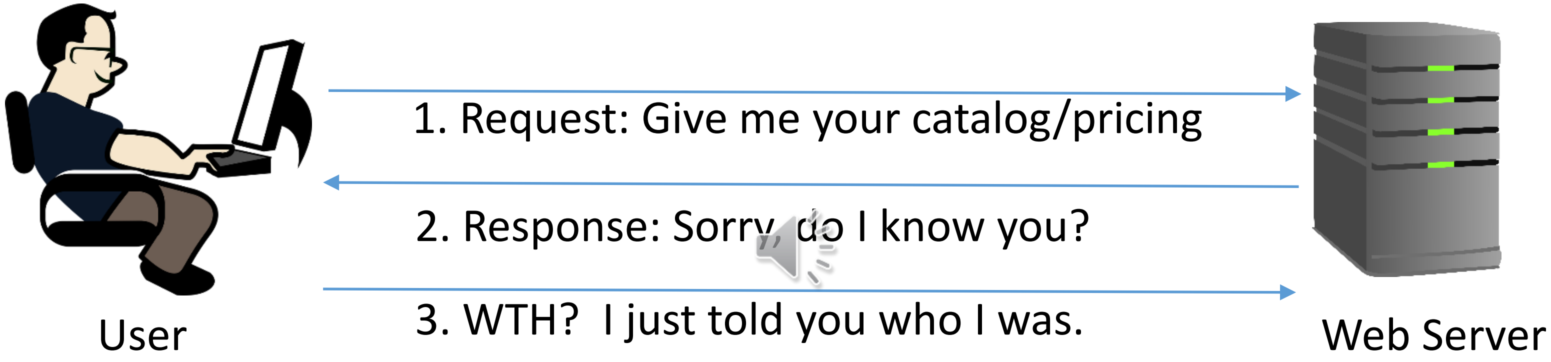
1. Request: Here is my user id and password.

2. Response: Ok! Welcome Mr. Boeing!.

User

Web Server

Web protocol – Post login request



1. Request: Give me your catalog/pricing

2. Response: Sorry, do I know you?

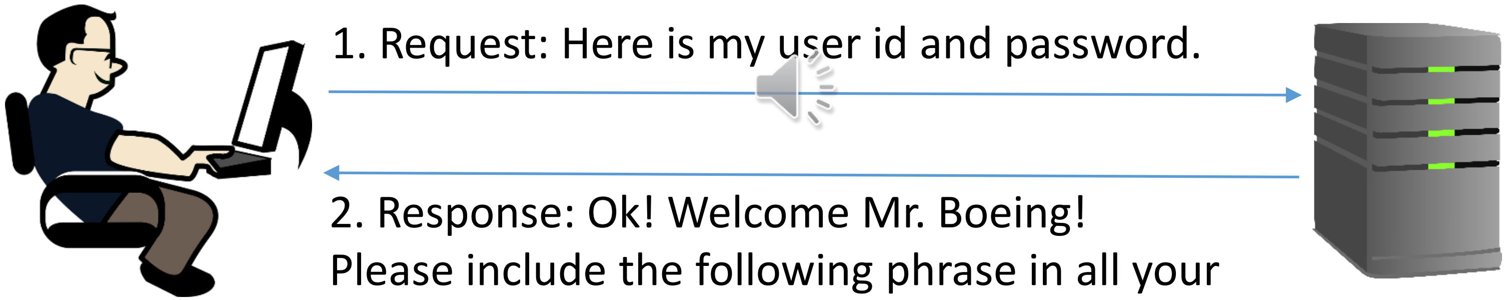
3. WTH? I just told you who I was.

User

Web Server

- Web server has no idea of what the user did before.
- The user has to remind the webserver, every time (every request) who he is, because web is stateless.

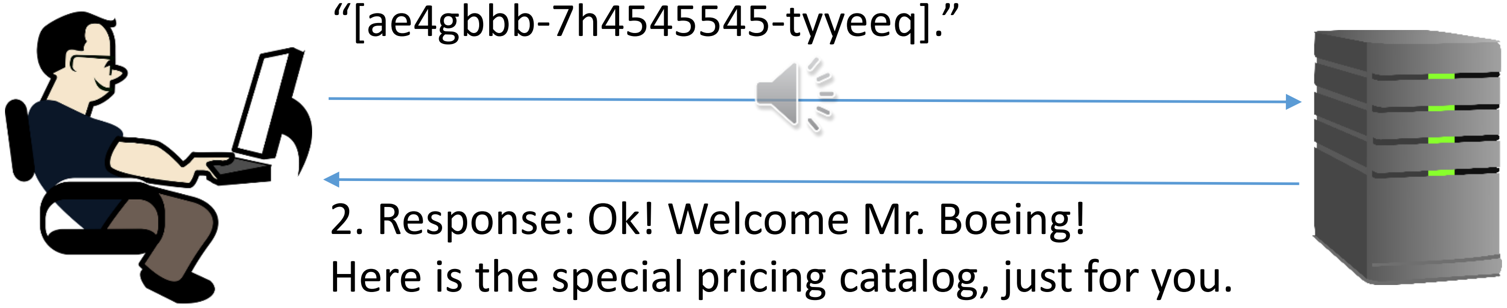
Web protocol – A better Login process



1. Request: Here is my user id and password.

2. Response: Ok! Welcome Mr. Boeing!
Please include the following phrase in all your future messages:
“[ae4gbbb-7h4545545-tyyeeq].”

Web protocol – Subsequent requests



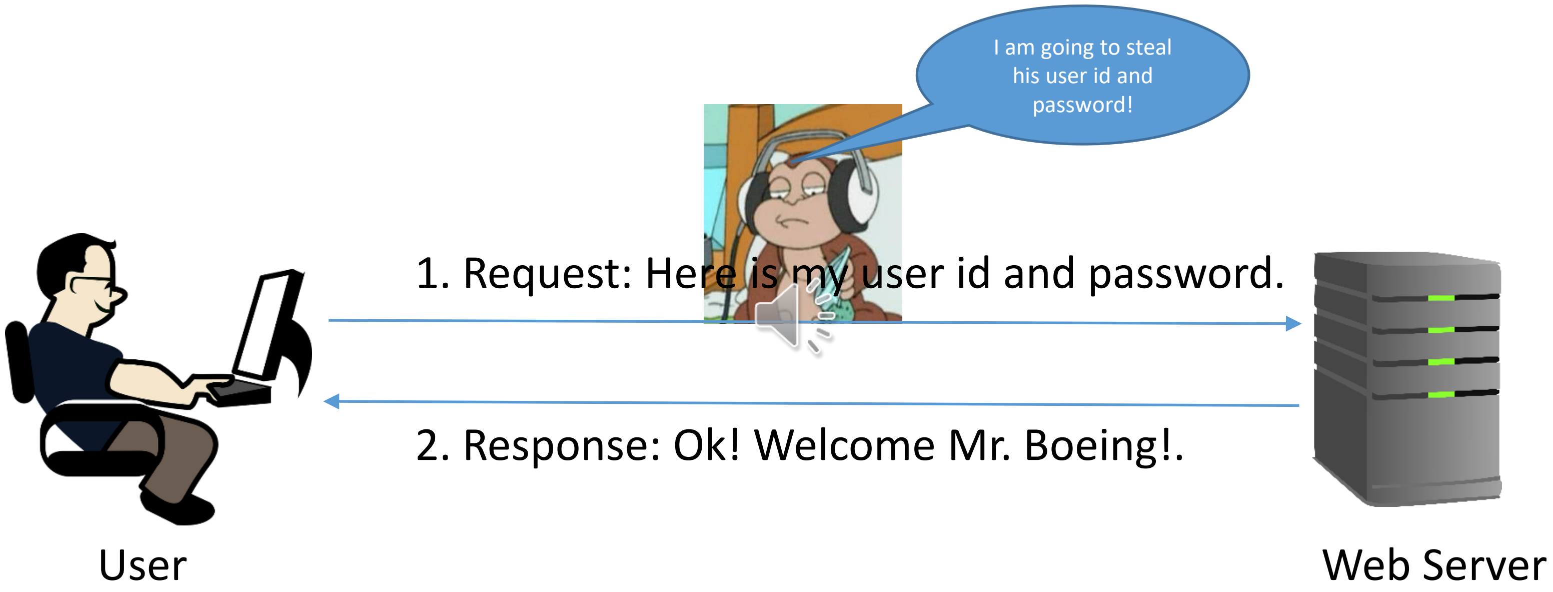
1. Request: Give me your catalog/pricing.
Oh by the way you asked me to include this phrase in every message:
“[ae4gbbb-7h4545545-tyyeeq].”

2. Response: Ok! Welcome Mr. Boeing!
Here is the special pricing catalog, just for you.

Web protocol – Lessons learned

- Web protocol is stateless. It does not remember any past conversations. The user has to jog the web server's memory about its past interactions, if any.
- The phrase sent by the server, is referred to as the token. It works almost like the hand stamping at an event, where once your hand is stamped, you don't have to take out your ID every time you go in and out of the venue.
- If you understood the previous slides without me using any jargons, then you know the stuff is really simpler than what it has been made to be.

Web protocol – The monkey in the middle



1. Request: Here is my user id and password.

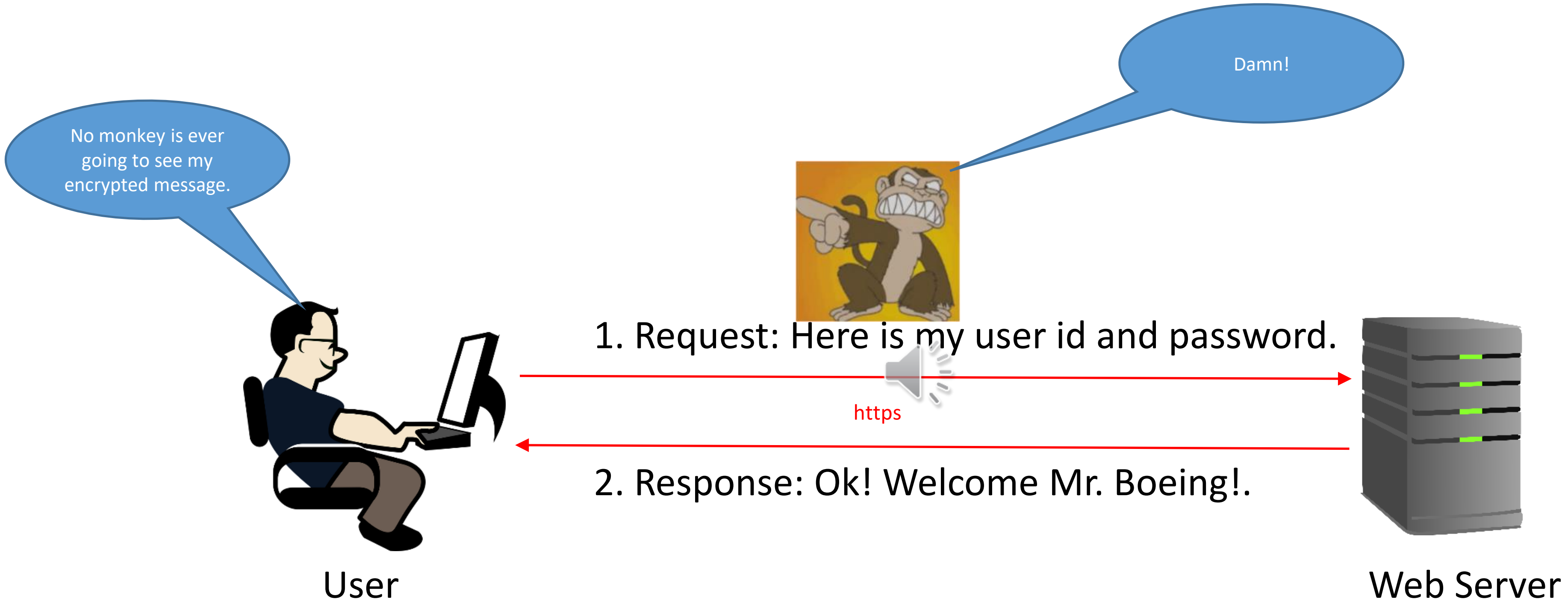
2. Response: Ok! Welcome Mr. Boeing!.

User

Web Server

The Evil monkey: Courtesy Family Guy

Web protocol – The monkey in the middle



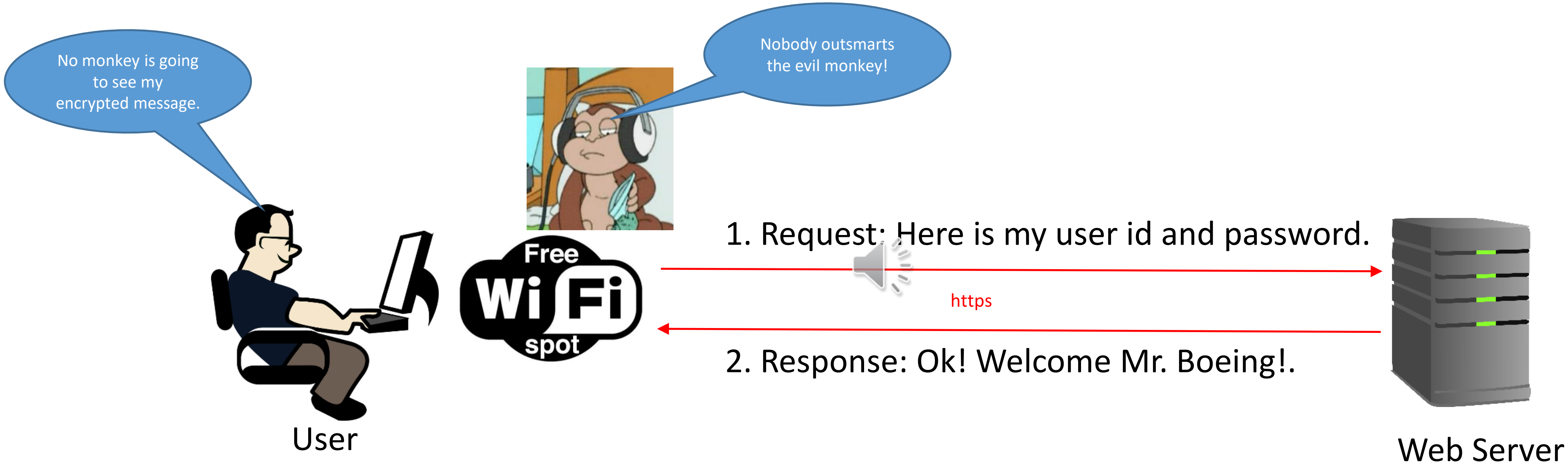
1. Request: Here is my user id and password.

2. Response: Ok! Welcome Mr. Boeing!.

Don’t run your site without https.

The Evil monkey: Courtesy Family Guy

Web protocol – Monkeys are everywhere



No monkey is going to see my encrypted message.

Nobody outsmarts the evil monkey!

1. Request: Here is my user id and password.

https

2. Response: Ok! Welcome Mr. Boeing!.

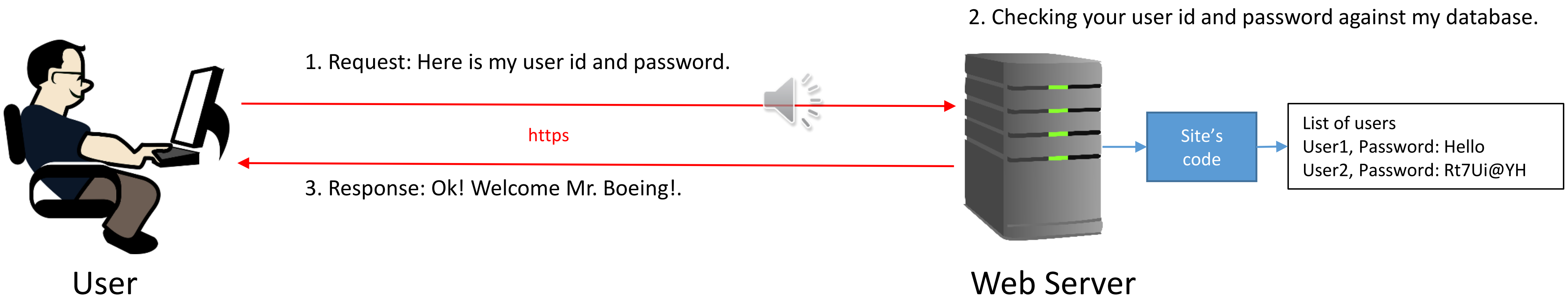
User

Web Server

The Monkey wins! Mr. Boeing should not have used the public Wi-Fi for important matters. Even https is prone to hacking under unknown networks.

The Evil monkey: Courtesy Family Guy

Let us get a bit technical – Login process



2. Checking your user id and password against my database.

1. Request: Here is my user id and password.

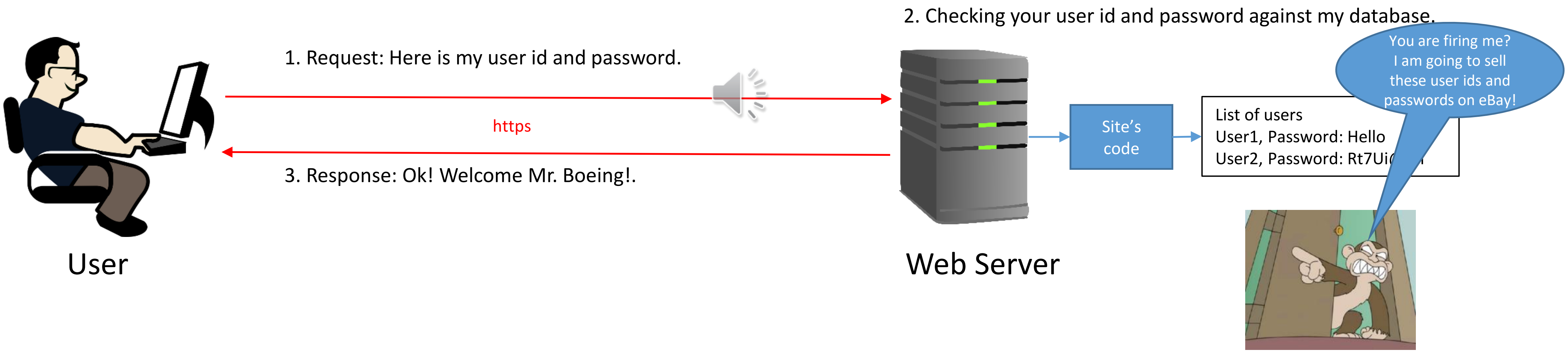
https

3. Response: Ok! Welcome Mr. Boeing!.

Site’s code

List of users
User1, Password: Hello
User2, Password: Rt7Ui@YH

Monkeys are still everywhere



2. Checking your user id and password against my database.

1. Request: Here is my user id and password.

https

3. Response: Ok! Welcome Mr. Boeing!.

User

Web Server

Site's code

List of users
User1, Password: Hello
User2, Password: Rt7Ui(

The Monkey wins! You, Mr. Developer should NOT have stored passwords in plain text!
Store only one way encrypted passwords so that they cannot be decrypted if the information was compromised.

The Evil monkey: Courtesy Family Guy

Let us get a bit technical – What does a typical web message look like

A web message is a long string of characters divided into two logical parts.

Headers

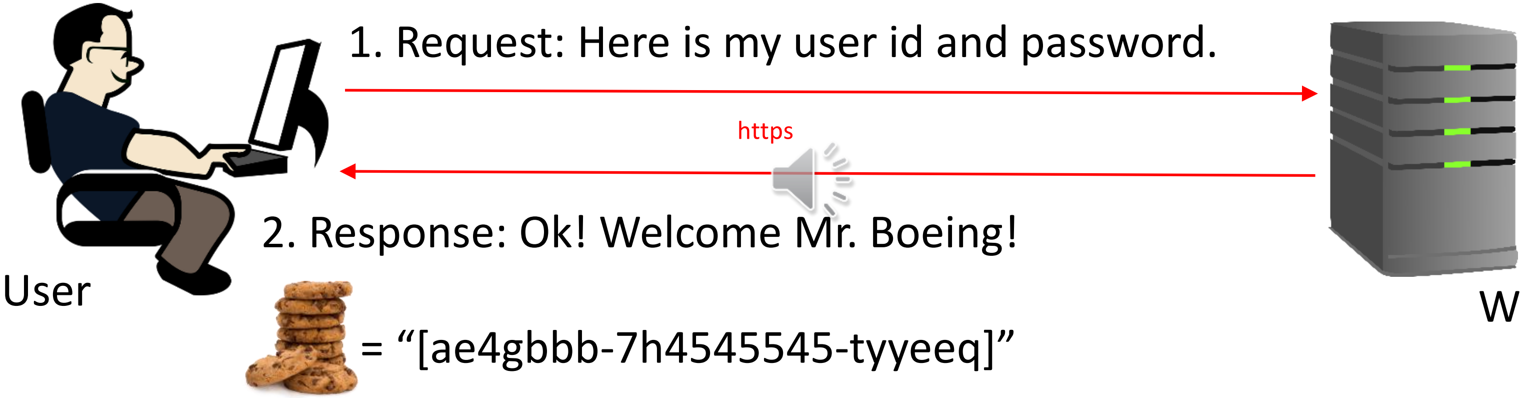
A lot of important stuff, but completely transparent to the user.

Body

Good stuff, the user sees on the page as well as form data posted by the user.

The cookies are one of the ways to restore state in a stateless environment.
They are stored in the browser and get saved on the user's hard drive.

Sending the token using cookies



1. Request: Here is my user id and password.

https

2. Response: Ok! Welcome Mr. Boeing!

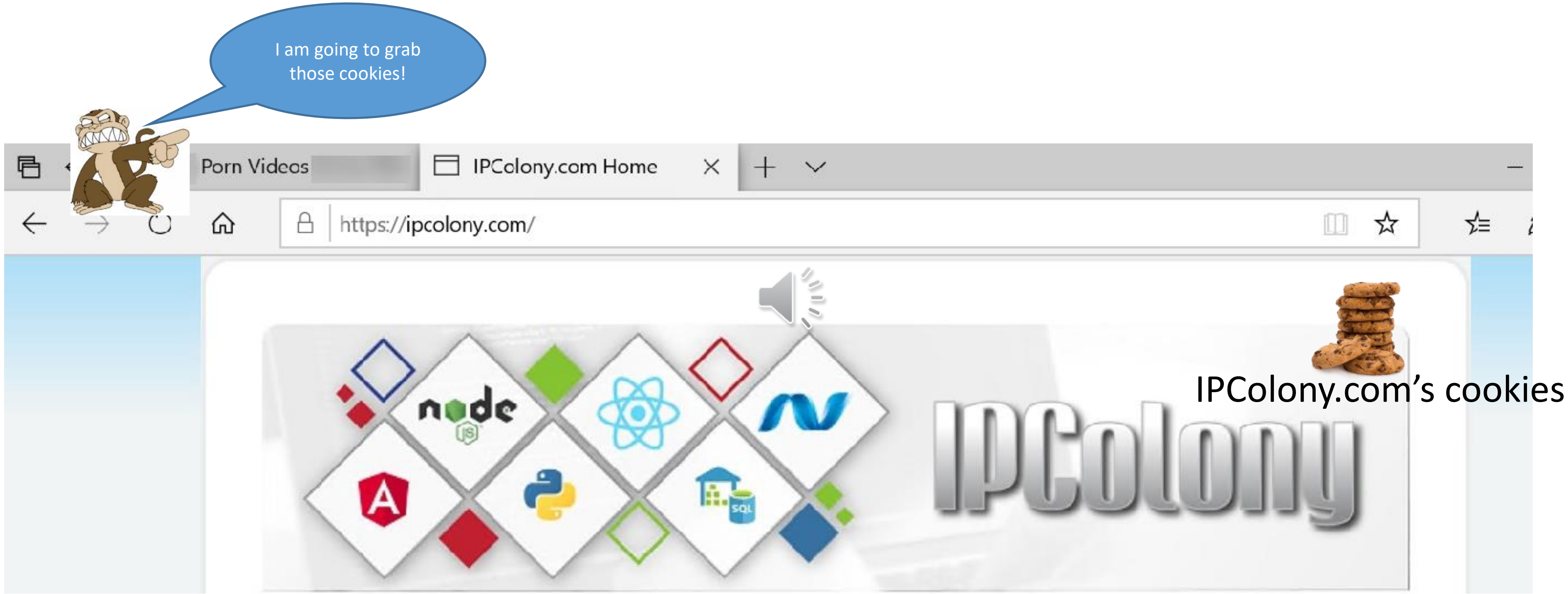
User

= “[ae4gbbb-7h4545545-tyyeeq]”

Web Server

Looks great so far! But the evil monkey is coming.

The monkey steals the cookies



A malicious script running from the other browser tab has equal access to another site's cookies and can make use of it.
Avoid using cookies to store sensitive information, use http headers instead.
Secure your site to not accept messages from another domain.

The Evil monkey: Courtesy Family Guy

Lessons learned

- The cookies are not the best places to hide secret information as they are susceptible to malicious scripts running from the other tabs of a browser.
- Use of cookies does not require any extra coding effort as they are always transmitted to the server.
- A secured solution is to store information in such a way that one browser tab does not know about the second one. This can be achieved by keeping the secret information in browser session’s memory. This poses a small challenge. You have to write code to inject this secret information (Example: session token) in every call made to the server.

Summary

- Be mindful, the web is a stateless environment.
- Use https instead of http.
- Use token based exchanges after logins.
- Store only one way encrypted passwords in the database.
- Prefer request headers over cookies.
- Don’t try to write your own encryption algorithms. Leverage the available work already done by much smarter people.

Next Step
Download the code of my compact ASP.Net application which covers many of these concepts.
https://ipcolony.com/was